**מערכת נוירונים ותיוג נתונים באמצעות פרספטרון**

**יחזקאל חי פדידה**

**רגרסיה לוגיסטית-** הוא אלגוריתם לסיווג מבוקר שפותר את הבעיה הקלאסית של למידה מקוונת של מערכת על נתונים. אלגוריתם זה מסווג נתונים בינארי (כל איבר מסווג לאובייקט אחד בדיוק מבין שתיים). כלומר זהו אלגוריתם סיווג שהתחזיות שלו מתבססות על פונקציית מנבא ליניארי שהיא צירוף ליניארי של ווקטור משקלות עם ווקטור המאפיינים. האלגוריתם מאפשר למידה מקוונת, בכך שהוא מעבד את הדוגמאות באימון אחת-אחת.

בשביל שאלגוריתם כזה יעבוד עלינו תמיד להחזיק ווקטור **איברים במערכת** ווקטור **משקלים** וקטור המשקלים הוא וקטור שמבדיל בין עדיפות לכל נתון באיבר אותו אנו רוצים לסווג. עלינו להחזיק משתנה "חיזוי" (threshold) הנקבע על ידי המשתמש. המשתנה הזה יקבע את הגבול של מה הוא איבר מסוג א' ומה הוא איבר מסוג ב'.

**מדוע נשתמש ברגרסיה לוגיסטית על פני רגרסיה ליניארית**

בשביל לפתור בעיית סיווג נרצה להשתמש בפונקציה h המקיימת . בעוד שברגרסיה ליניארית פונקציית המטרה היא מהצורה  *.* שכן היא מכילה הרבה נקודות מינימום מקומי (אינסוף), כלומר היא לא פונקציה קמורה. ולכן, נעשה שימוש ברגרסיה לוגיסטית מהצורה הבאה (פיתוח נוסחה בהמשך) מה שיבטיח מציאת מינימום גלובלי.

**פיתוח מתמטי של האלגוריתם**

נתייחס לרגרסיה לוגיסטית כאל פונקציה המוגדרת בצורה הבאה

נסמן באופן כללי ונסמן את אוסף כל הדאטה שלנו בצורה הבאה  *כאשר x הוא וקטור הערכים עבור כל איבר בדאטה ו y הוא הסיווג האמיתי (אפס או אחד). בנוסף עלינו להגדיר וקטור משקולות למערכת לכן נסמן כוקטור המשקולות (וקטור המשקולות w ווקטור האיברים x יהיה מאותו המימד מכיוון שלכל ערך קיימת משקולת)*

***המטרה: למצוא את וקטור המשקולות שימקסם (max) את ההסתברות שהמערכת תהיה צודקת***

*נסמן את ההסתברות של y בהנתן x ו- w*

כעת יהיה עלינו למקסם את עבור כל i . מיקסום ערך זה הוא שווה ערך לערך הבא

*(זאת מכיוון שהסתברות P היא ערך חיובי כלשהו)*

***שימוש בפונקצית סיגמואיד***

נניח שאת הההסתברות ניתן לקרב באמצעות פונקציה כלשהי. לכן נגדיר

*מכיוון ש יכול להיות כל מספר ב R נצרך למצוא פונקציה F כך שמתקיימים התנאים הבאים זאת מכיוון שאנו רוצים למצוא הסתברות כלשהי והסתברות כלשהי היא ערך ששייך לקטע [0,1] . פונקציה פופולרית ברגרסיה לוגיסטית המקיימת את התנאים הללו היא פונקציית סיגמואיד. פונקציית סיגמואיד נראית כך*

נגדיר את פונקצית המטרה הבאה: M(w)=

ונפתח אלגברית את הערך הבא

*שימוש בפונקצית סיגמואיד*

ניתן להציג את ערך זה בצורה הבאה-

בשביל להמשיך בתהליך הפיתוח של המשוואה ננסה להמיר את המשוואה לחיבור של n איברים במקום מכפלה של n איברים. נשים לב שאם נפעיל לוג על המשוואה נוכל לפשט אותה אלגברית ביעילות. נציין כי הפעלת לוג על הפונקציה שקיבלנו לא תשנה את ערך המקסימום של w

**קיבלנו את הפונקציה נפעיל את שיטות האופטימיזציה בשביל לקבל את הערכים האידיאלים עבור וקטור המשקולות w .**

**נרצה למצוא את המינימום עבור פונקצית ה Loss לכן כפי שלמדנו בהרצאות ניתן להוציא מינוס לכל הפונקציה וערך המינימום של פונקצית המינוס יהיה שווה לערך המקסימום של הפונקציה הנ"ל. סה"כ קיבלנו את הפונקציה הבאה**

**Loss function**

**שיטת גרדיאנט למציאת המינימום**

בהינתן וקטור w מהצורה הבאה

על מנת למזער את השגיאה של פונקצית המטרה Loss ערך המשקולות המעודכן יצטרך להיות

כך שאלפא הוא קצב הלימוד (learning rate).

ראשית נחשב נגזרת לפי x של פונקצית סיגמואיד

\*

כעת נחשב את גרדיאנט של פונקצית המטרה שלנו w(x)

כעת נשתמש בכלל השרשרת בו "מצמצמים" דיפרנציאלים

לפי נגזרת של לוגים מתקיים

נשתמש שוב בכלל השרשרת

נשתמש בחוק שמצאנו \* ונקבל

נצמצם ונקבל

נוציא אל מחוץ לסוגריים ונקבל

*נפתח סוגריים ונצמצם*

***לבסוף נקבל כי***

**אופטימיזציה קמורה ולמה השתמשנו בשיטת גרדיאנט?**

קיימת לנו כאן בעיית אופטימיזציה קמורה. אופטימיזציה קמורה היא תת תחום באופטימיזציה שמתעסקת בבעיות שפונקציית המטרה שלהן היא פונקציה קמורה. במקרה זה של רגרסיה לוגיסטית חשוב להבחין שקיימת בעיית אופטימיזציה קמורה ולנצל את התכונות של פונקציה קמורה (כל מינימום לוקאלי של פונקציה יהיה המינימום הגלובלי שלה). במקרה שלנו נרצה למצוא את ערכי המשקולות המתאימים ביותר בשביל למצוא את המינימום הגלובלי של ערכי "ההפסד" של המודל. אם נוכיח שהפונקציה הינה קמורה נוכל להבטיח ששיטת הגרדיאנט תמצא לנו את הערכי המשקולות המתאימים! לכן עלינו להוכיח כי פונקציית המטרה שמצאנו הינה קמורה.

נזכר כי הפונקציה שמצאנו היא הפונקציה הבאה

נבצע מניפולציה אלגברית

כעת נסתכל על הביטוי הבא שנמצא בסיגמא

נפתח אלגברית ונקבל

**נוכיח קמירות**

לפי הפיתוחים האלגבריים שביצענו קיבלנו את המשוואה הבאה

אשים לב כי החלק הראשון בסיגמא הינו **ליניארי!** לפי הגדרה פונקציה ליניארית הינה קמורה.

לכן אם נראה כי החלק השני של הסיגמא גם קמור אז נוכל להבטיח שפונקציית המטרה שלנו כולה קמורה. נראה כעת כי החלק השני גם הוא קמור.

נחשב גרדיאנט עבור הפונקציה

**כעת יהיה עלינו לחשב את ההסיאן**

ההסיאן במקרה שלנו יהיה שווה לנגזרת השנייה של הגרדיאנט שמצאנו לפי w.

נמצא את ההסיאן

כעת נראה כי המטריצה מוגדרת חיובית.

יהי V וקטור כלשהו אזי עבור על V מתקיים

קל לראות כי ביטוי זה יהיה תמיד גדול שווה מאפס. התמונה של היא ערך בין אפס לאחד לכן הביטוי הינו גדול שווה מאפס. אם נכפול ביטוי זה בנורמה בריבוע נקבל בכל מספר חיובי לכן ניתן להסיק כי

קיבלנו כי ההסיאן של מוגדרת חיובית ולכן נסיק כי פונקציה זו הינה קמורה.

כעת ניתן להסיק כי פונקציית ההפסד

הינה קמורה לכן אם נשתמש בשיטת גרדיאנט נוכל להבטיח כי נמצא את המינימום הגלובלי שלה.

**שימושים שונים של האלגוריתם**

באמצעות אלגוריתם זה ניתן ללמד מחשב להבדיל בין סוגים שונים של נתונים. האלגוריתם הזה הוא בסיס לאלגוריתמים רבים שאני משתמשים בהם ביום יום מבלי לשים לב. אחת הדוגמאות הנפוצות לשימוש באלגוריתם הזה הוא סיווג של דואר אלקטרוני. לכל תיבת דואר יש תיקיית ספאם, אפשר להגיד שמאחורי הקלעים פועל אלגוריתם שכזה עם כמות עצומה של דאטה ובעזרת שיטות מתמטיות תיבת הדואר שלנו תדע לזהות מה הוא דואר זבל ומה הוא דואר רגיל.

בעבודה זו אממש את בסיס האלגוריתם ואציג כיצד ניתן לאמן נתונים שונים בשביל הצרכים האישיים שלנו, אבנה אלגוריתם שמקבל קבוצת בקרה של דאטה כשהמטרה העיקרית שבסוף הרצת האלגוריתם המחשב ידע להבדיל באופן בינארי בין סוגים שונים של דאטה.

**פיתוח האלגוריתם**

שלב א

ראשית נקבע מה יהיה משתנה "החיזוי" שלנו. בשביל לבדוק באופן שיטתי מה הוא סוג האיבר אותו אנו רוצים לסווג יהיה עלינו להכפיל את וקטור המאפיינים של האיבר הנתון בוקטור המשקלים. את התוצאה שנקבל נצטרך להשוות למשתנה החיזוי. במידה ומכפלת הוקטורים תהיה קטנה ממשתנה החיזוי נגדיר את האיבר כסוג א' (0) ובמידה ויהיה גדול ממנו נגדיר את האיבר כסוג ב' (1)

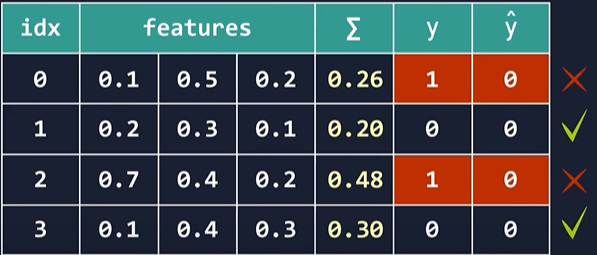
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שלב ב

בשביל "לאמן" את המערכת שלנו ולהגדיר מה היא תוצאת סיווג טובה ומה היא תוצאת סיווג שנכשלה עלינו להגדיר עבור כל איבר משתנה בינארי מסוג "מטרה" במשתנה זה נחזיק עבור כל איבר קיים שאנו כבר כמשתמשים יודעים לסווג מאיזה סוג המשתנה, א' או ב'. ובכל פעם שנריץ את האלגוריתם נבדוק האם משתנה "החיזוי" שלנו אכן תואם למשתנה ה"מטרה"

דוגמא להמחשה-

 כאשר הוא משתנה המטרה ו y הוא משתנה החיזוי של המערכת.

שלב ג

עלינו לבחור "פונקציית הפסד", משמעות פונקציית ההפסד שלנו היא עד כמה המודל שלנו צודק. ככל שהערך של פונקציית ההפסד (loss) נמוכה יותר כך המודל "צודק" יותר.

נבחר את loss function שלנו

נחשב עבור כל איבר שנרצה לסווג את "ההפסד" שלו כלומר כמה החיזוי קרוב למטרה לפי המודל המתמטי שפיתחנו בתחילת העבודה קיבלנו כי פונקציית ההפסד שלנו תהיה

נחשב Total loss על ידי חישוב "ההפסד" של כל איבר שבדקנו ונחלק במספר האיברים (כלומר ממוצע של הסטייה של המערכת)

סך הכל נקבל

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיזוהי פונקצית ה loss הסופית שלנו שלפיה נעבוד (מותר לנו לחלק ב n מכיוון שמספר זה הינו קבוע!)



ניתן להסיק בעיית האופטימיזציה שנתעסק בה היא בעיית מינימום כאשר הפונקציה אותה נרצה להביא לערך הקטן ביותר (מינימום) היא פונקציית ההפסד שלנו (הבאת פונקציה זו לערך מינימלי תמזער לנו את אחוזי השגיאה של המערכת כלומר המערכת תהיה יותר מדויקת)

**כיצד נבצע אופטימיזציה?**

**לשם כך נגדיר את המושגים הבאים-**

**Threshold**

לפי ערך זה אנו כמפתחי המערכת נוכל לסווג כל איבר. נצטרך לבחור threshold שלפיו נסווג. כלומר אם מתקיים weighted sum < threshold אז נסווג את האיבר ל-0 אחרת נסווג את האיבר ל 1. נבחר threshold בדרך כלל לפי הצורך של המערכת שאותה אנו מפתחים (התפלגות קיצונית לאחד הכיוונים, התפלגות סטנדרטית ועוד..) בדרך כלל נבחר threshold=0.5 סטנדרטי.

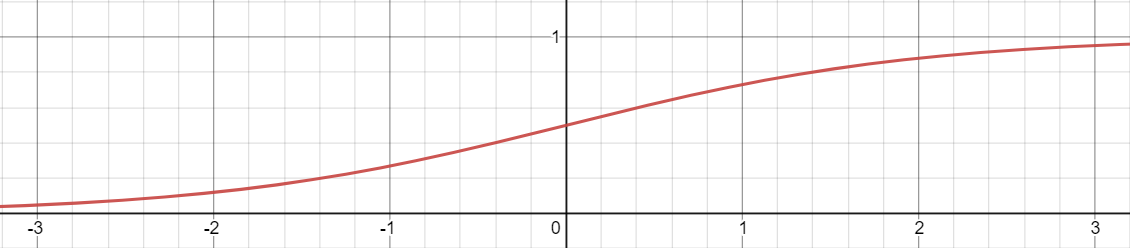
**Weighted sum (סכום המשקלים)**

לכל איבר שמופיע ב data שלנו קיים ערך בשם weighted sum ערך זה הוא סקלר והוא ערך המכפלה הוקטורית בין וקטור המשקלים של המערכת לוקטור הערכים של האיבר הספציפי.

**Sigmoid**

פונקציית סיגמואיד היא פונקציה מתמטית, בעלת עקומה בצורת האות "S" הנקראת גם עקומת סיגמואיד.

הפונקציה



פונקציית סיגמואיד הינה פונקציה ממשית דיפרנציאבילית מונוטונית עולה וחסומה המוגדרת על המספרים הממשיים ובעלת נגזרת אי שלילית בכל נקודה בה. לפונקציה זו נגדיר את המקורות כסכום וקטור המשקלים ונקבל ערך Y בין 0 ל 1. בעזרת ערך זה נוכל לשנות את המשקולות בהתאם לצרכים שלנו ונוכל לסווג ביותר קלות. פונקציה זו הינה פופולארית מאוד ברגרסיה לוגיסטית מכיוון שהיא יכולה לתת ערכים רק בין 1 ל 0 בדיוק טווח הערכים להסתברות כלשהי (ההסתברות שאיבר מסוים מסווג כ- 1 או כ- 0). **במשך כל התוכנית נגדיר את משתנה החיזוי של המערכת שלנו כ f(weighted sum) כאשר f היא סיגמואיד.**

**Loss (הפסד) עבור כל איבר**

הרי משתנה החיזוי שלנו עבור כל איבר הינו f(weighted sum) כך ש f סיגמואיד. נגדיר ערך זה כמשתנה prediction ונגדיר את פונקציית ההפסד הלוגריתמית הבאה.

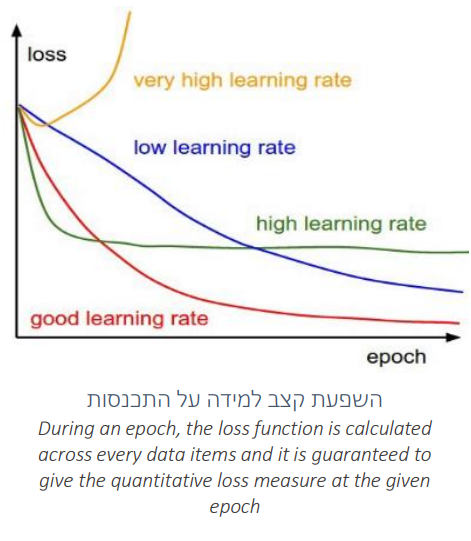
**Gradient Decent**

אחת משיטות האופטימיזציה המוכרות ביותר. בשיטה זו אנחנו מחשבים את כיוון הגרדיאנט מהנקודה שבה אנחנו נמצאים ברגע הנתון (משקלי הרשת) בהתאם לפונקצית ההפסד. ומתקדמים מהנקודה הנוכחים בצעד קטן בכיוון הנגדי לכיוון הגרדיאנט (כיוון הגרדיאנט הוא הכיוון בו ערך הפונקציה עולה בצורה מקסימלית, הכיוון הנגדי הוא הכיוון בו הפונקציה יורדת בצורה מקסימלית כלומר מינימום) נשתמש בשיטת הגרדיאנט על הפונקציה שתייצג את Loss הרי פונקציה זו מייצגת לנו כמה המערכת שלנו מסווגת בצורה טובה וכמה היא מדייקת בממצאים שלה.

על מנת למזער את ערך השגיאה נעדכן את ערכי המשקולות בצורה הבא

**Learning rate**

אפשר להתייחס אל קצב הלמידה כאל הצעד בשיטת הגרדיאנט. עלינו לבחור קצב למידה אופטימלי מכיוון שנתונים יכולים להיות בלתי צפויים, נצרך למצוא קצב למידה שיכסה את כל הנתונים ולא יפספס נתונים חשובים. הדרך למציאת קצב למידה אופטימלי הינה בדרך של ניסוי וטעיה.



***Epoch***

*"הלימוד" של המערכת בפועל נעשה במסגרת של מחזורי למידה. המושג Epoch מתייחס למחזר למידה בודד. בסוף כל מחזור למידה פונקציית ה Loss מעריכה את דיוק המודל לפי הדוגמאות בקבוצת הבקרה. על סמך ערך זה פונקציית האופטימיזציה מעדכנת את המשקולות.*

***פעולת האימון Train***

***שלב א אימון עבור איבר יחיד***

1. *נחשב עבור איבר יחיד את ערך weight sum שלו (מכפלה וקטורית של וקטור המשקלים ווקטור הערכים)*
2. *נחשב את הערך Pred על ידי הפעלת פונקציית סיגמואיד על הערך weight sum ערך זה יהיה הערך שלפיו המערכת תבצע סיווג בינארי*
3. *נחשב את ערך Loss עבור האיבר הספציפי. נכניס את ערך Pred אל פונקציית Loss שלנו ונחשב את הערך המתאים (אחוזי הדיוק).*
4. *נשתמש בשיטת גרדיאנט ונעדכן את ערכי המשקלים בהתאם לנתונים בשביל למזער את אחוז השגיאה.*
5. *נשמור את ערך השגיאה של האיבר הספציפי*

***שלב ב השלמת Epoch***

1. *לפי ההגדרה שהבאנו נבצע את שלב א כמספר האיברים הנתונים בקבוצת הבקרה*
2. *כאשר סיימנו Epoch שלם נחבר את כל ערכי השגיאה העכשוויים של האיברים בקבוצת הבקרה ונחלק במספר האיברים (ממוצע הפסד)*

*אם הקוד תקין ככל שנאמן יותר את קבוצת הבקרה נצפה לראות ירידה משמעותית של ממוצע ההפסדים הכללי!*

***כאשר המודל מוכן***

*כאשר המודל מוכן לאחרי אימונים רבים של קבוצת הבקרה יוכל משתמש חיצוני להכניס איבר כלשהו ולפי הנתונים שהספיקה ללמוד המערכת. המחשב יוכל לסווג אותו לפי ה threshold שבחרנו!*

***פונקציות פייתון***

***Get weighted sum***

*פונקציה זו מקבלת את האיבר הספציפי עבורו נרצה לחשב את הערך Weighted sum. הפונקציה תבצע מכפלה של וקטור הערכים בוקטור המשקולות ותחזיר לנו את הערך של Weighted sum*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

***Sigmoid***

*פונקציה זו מקבלת את ערך Weighted sum ותחזיר את הערך*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

***Loss calc***

*פונקציה זו מקבלת את ערך מטרה ספציפי Target ותקבל בנוסף ערך Prediction עבור אותו האיבר. הפונקציה תחזיר את הערך הבא:*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

**Calc\_acurrity**

*פונקציה זו מקבלת את מספר האובייקטים שקיימים בדאטה, וקטור המשקולות ואת מערך האובייקטים. הפונקציה סכמת את אחוזי הדיוק של המערכת ברגע הנתון עבור כל אובייקט בדאטה. לבסוף היא מחלקת במספר איברי הדאטה (לקבל ממוצע) ומדפיסה את אחוזי הדיוק של המערכת. בעזרת פונקציה זו נוכל לכמת את העבודה של פונקצית train.*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

***Gradient decent***

*הפונקציה מקבלת את וקטור הערכים עבור איבר ספציפי, וקטור משקולות, המטרה Target, צפי המערכת Predection , וקצב הלמידה. הפונקציה משתמשת בשיטת גרדיאנט ומחזירה לנו את וקטור המשקולות המעודכן*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

**Train data**

*הפונקציה מקבלת מערך הכולל את כל האיברים הקיימים Features, וקטור המשקולות, וקטור המטרות Targets, מספר האיברים בפועל ומספר ה Epochs שהמשתמש בוחר לבצע.*

*הפונקציה מאמנת את האיברים Epochs פעמים, ולבסוף מחזירה את ערכי המשקולות המעודכנים ומחזירה מערך המכיל את כל ערכי ה Loss לאורך התהליך כאשר האיבר האחרון במערך הוא ערך ה Loss המעודכן.*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

***איך אנו מבטאים בעצם את המתמטיקה בפונקציות פייתון?***

***פונקצית loss***

*בחישובים המתמטיים שהצגנו בעמודים הקודמים קיבלנו כי פונקציית ה Loss שלנו היא מהצורה הבאה*

*אשים לב שלפי מה שהגדרנו לאורך התהליך הוא הפעלת פונקצית סיגמואיד על ערך weighted sum ערך זה הוא בעצם הערך חיזוי של המערכת עבור האיבר הספציפי נסמן =pred*

*בנוסף לפי איך שהגדרנו הערך הינו הערך המקורי של האיבר שכבר קיים בקבוצת הבקרה. נסמן Target=*

*בפונקצית הגרדיאנט שלנו אנו מוסיפים את כל ערכי ה loss calc למערך. ולבסוף אנחנו סוכמים את כל הערכים שקיבלנו מה שנותן לנו בדיוק את ערך הפונקציה שקיבלנו לפי החישובים המתמטיים (לבסוף אנו מחלקים ב n מטעמי נוחות)*

***פונקצית gradient decent***

*בחישובים המתמטיים שהצגנו בעמודים הקודמים קיבלנו כי נעדכן את המשקלים בשיטת גרדיאנט בצורה הבאה*

*אשים לב שהרי הביטוי הינו הביטוי target-predection*

*אשים לב בנוסף כי הינו הביטוי feature[i]*

בכל איטרצית epoch יחידה נבצע הפעולה בדיוק n פעמים (כך בעצם מתבטאת הסיגמא)

***קלט ופלט***

***קלט***

*המשתמש בוחר learning rate בקצב של 0.2, בנוסף הוא בוחר שיהיו 4 איברים בקבוצת הבקרה כך שכל איבר מאופיין על ידי 3 גורמים שונים. בנוסף מוגרל וקטור משקולות אשר מגדיר ערך משקל עבור כל מאפיין. הטבלה בנוסף מראה מאיזה סוג מוגדר כל איבר (אפס או אחד)*

***תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי***

***תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי***

***פלט***

*בהרצה זו המשתמש בחר Epoch=10 כלומר פעולת האימון תתבצע 10 פעמים! במערכת מראה עבור כל Epoch מה הוא ממוצע Loss ברגע הנתון.*

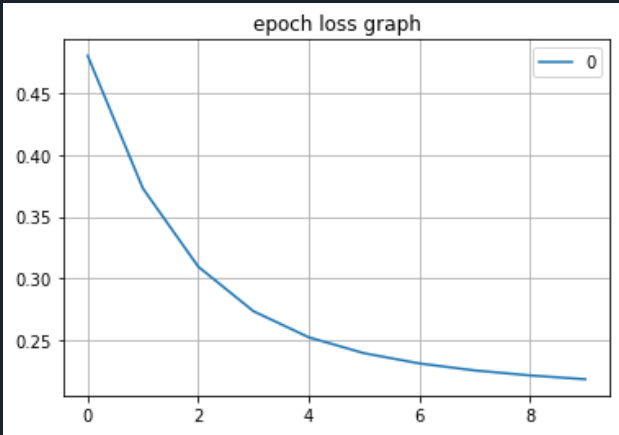
***תמונה שמכילה טקסט, אלקטרוניקה

התיאור נוצר באופן אוטומטי***

*המערכת מחזירה את וקטור המשקולות העדכני עבור ממוצע ההפסד ובנוסף מחזירה גרף אשר מראה את קצב ההתכנסות לאפס (מערכת זיהוי מושלמת) עבור מספר ההרצות. המערכת בנוסף תדפיס את אחוז הדיוק העדכני.*

*תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי*

******

***ביבליוגרפיה***

❖

[***https://zephyrnet.com/iw/demystification-of-logistic-regression/***](https://zephyrnet.com/iw/demystification-of-logistic-regression/)

❖

[***https://status.co.il/%D7%9E%D7%95%D7%93%D7%9C%D7%99%D7%9D-%D7%9C%D7%99%D7%A0%D7%90%D7%A8%D7%99%D7%99%D7%9D-%D7%91%D7%91%D7%A2%D7%99%D7%95%D7%AA-%D7%A1%D7%99%D7%95%D7%95%D7%92-%D7%A8%D7%92%D7%A1%D7%99%D7%94-%D7%9C%D7%95/***](https://status.co.il/%D7%9E%D7%95%D7%93%D7%9C%D7%99%D7%9D-%D7%9C%D7%99%D7%A0%D7%90%D7%A8%D7%99%D7%99%D7%9D-%D7%91%D7%91%D7%A2%D7%99%D7%95%D7%AA-%D7%A1%D7%99%D7%95%D7%95%D7%92-%D7%A8%D7%92%D7%A1%D7%99%D7%94-%D7%9C%D7%95/)

❖

[***https://he.isecosmetic.com/wiki/Logistic\_regression***](https://he.isecosmetic.com/wiki/Logistic_regression)

[***https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a***](https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a)

❖

[***https://he.wikipedia.org/wiki/%D7%A8%D7%A9%D7%AA\_%D7%A2%D7%A6%D7%91%D7%99%D7%AA\_%D7%9E%D7%9C%D7%90%D7%9B%D7%95%D7%AA%D7%99%D7%AA#%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D\_Gradient\_Descent***](https://he.wikipedia.org/wiki/%D7%A8%D7%A9%D7%AA_%D7%A2%D7%A6%D7%91%D7%99%D7%AA_%D7%9E%D7%9C%D7%90%D7%9B%D7%95%D7%AA%D7%99%D7%AA#%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D_Gradient_Descent)

❖

***https://ecowiki.org.il/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%99%D7%AA\_%D7%A1%D7%99%D7%92%D7%9E%D7%95%D7%90%D7%99%D7%93***